DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[The technical field to which invention belongs] This invention relates to the incircuit emulator which plans effectiveness of debugging in the development tool of the application software of a microcomputer.
[0002]
[Description of the Prior Art] Importance is attached to how the waiting power of an electric product is stopped in recent years. Therefore, when realizing the low power of an electric product, since the effect is large when the original oscillation of a microcomputer is stopped, in order to realize a low power, in addition to required processing of the microcomputer used for many electric products, there is a function (it is hereafter called a stop mode) to make one of the modes of operation of a microcomputer suspend a original oscillation. The flow of the actuation which debugs the application software using the stop mode with the conventional incircuit emulator (hereafter referred to as ICE) is shown in drawing 5 . This drawing 5 shows the condition when performing actuation by ICE interrupt processing in the stop mode of a user program. Moreover, drawing 6 (a) is a flow chart with which the main routine of the user program which has the stop mode which debugs, and drawing 6 (b) show interrupt processing of a user program, and drawing 6 (c) shows ICE interrupt processing.
[0003] In drawing 6 (a), (b), and (c) the main routine of the user program of a microcomputer and 2 1 interrupt processing of a user program and 3 ICE interrupt processing by which interruption of ICE will be performed by generating if actuation of stopping a main routine 1 is performed, The instruction with which 4 contains the initialization routine of a user program and which usually puts processing and 5 into a stop mode, The usual processing performed after the interrupt processing 2 of a user program generates 6 in a stop mode, The 1st compulsive breaking by generating of the ICE interrupt processing 3 which 7 makes usually stop a user program in the middle of processing 4, The 2nd compulsive breaking according generating of interrupt processing 2 to generating of the ICE interrupt processing 3 waiting in the inside of the stop mode by the instruction 5 to which 8 is put into a stop mode, The return instruction for returning 9 to a main routine 1 from the interrupt processing 2 of a user program, The ICE return instruction of the ICE interrupt processing 3 for making 10 rerun from the activation idle state of a user program and 11 It is usual interrupt processing which is made to suspend activation of a user program, is controlled by the control software of ICE by the ICE interrupt processing 3 performed, and performs debugging etc.
[0004] Drawing 7 is a state transition diagram in the main routine of drawing 6 (a), and the flow chart of ICE interrupt processing of drawing 6 (c), and drawing 8 is drawing showing the stack area which shunts status information according to generating of interrupt processing. In drawing 8 , as for 12, the shunting field of the flag status and 13 are the shunting fields of the various statuses, and shunting storing of the shunting field of the return address of a program and 14 is automatically carried out in these shunting fields at the time of interruption generating.
[0005] The example of concrete actuation of said conventional example is explained using drawing 7 and drawing 8 . First, external interruption of the 1st compulsive breaking 7 which stops a user program during the usual processing 4 of the main routine 1 in a user program is generated. It changes to interrupt processing of ICE by the ICE interrupt processing 3 of this 1st compulsive breaking 7. Shunting storing of the condition of a user program is carried out in the stack area shown in drawing 8 , respectively like [this ICE interrupt processing 3] the interrupt processing 2 of a user program. Moreover, in the control software of ICE, the content of the stack area by which shunting storing was carried out can be read, and a condition when a user program stops can be displayed with a monitor. And since ICE interrupt processing 3 other than a user program is performed to the user, it is visible to the condition that the user program stopped. Furthermore, rerun of a user program is executing the ICE return instruction 10 of the ICE interrupt processing 3, and the resource by which shunting storing was carried out in the stack area at the time of generating of the 1st compulsive breaking 7 is returned.

[0006] Next, although the ICE interrupt processing 3 of the 2nd compulsive breaking 8 occurs in the stop mode by the instruction 5 put into the stop mode of a main routine 1 and being changed to interrupt processing of ICE, since the flow of the actuation here to a halt of a user program is the same as that of said explanation, it is omitted, and explains the different portion.
[0007] Since a user program is a stop mode, and the oscillation has stopped, a stop mode is canceled by generating of the ICE interrupt processing 3, and an oscillation is started. Hereafter, the condition of oscillating is called normal mode. That is, ICE interrupt processing 3 is performed by normal mode, and becomes things. And since rerun of a user program stopped in the state of the stop mode in this actuation, it is necessary to make it the original stop mode.
[0008] However, if processing put into a stop mode before the ICE return instruction 10 in the ICE interrupt processing 3 is performed, during the ICE interrupt processing 3, it cannot keep in a stop mode as close, and cannot return correctly at the mode of operation in front of a user program. For this reason, when making it rerun in such a conventional case, the control software of ICE was taking out warning of "since it is among a stop mode, please make it normal mode." Therefore, specifying the start address which a user changes the register which controls a mode of operation by ICE, or is rerun, or resetting and rerunning was performed.
[0009]
[Problem(s) to be Solved by the Invention] In the incircuit emulator of such a configuration In the case of debugging which develops the application software which has the stop mode which makes a user program suspend the original oscillation of a microcomputer In order to consider as the original stop mode so that interrupt processing of ICE may be performed into the stop mode of a user program and a user program can be rerun after that, Even if it performs the instruction put into a stop mode before the ICE return instruction of ICE interrupt processing, before returning to the execute mode of a user program, it keeps in a stop mode as close during interrupt processing of ICE, and the problem that it cannot return to the original condition is.
[0010] Moreover, in order to reput into a stop mode automatically after processing of the control software of ICE, it was the translation which should just carry out a return to the address put into the stop mode in the user program, but even if it executed the instruction put into a stop mode, the next instruction might also be executed, and if the instruction was branch instruction, where it should return had the problem that decision was impossible.
[0011] This invention aims at offering ICE (incircuit emulator) which was excellent in the operability which can be returned to the condition before directing to solve the problem of said conventional technology, and a user's not having any constraint in the case of rerun of the user program after performing ICE interrupt processing to the user program stopped in the state of the stop mode and stopping a user program.
[0012]
[Means for Solving the Problem] In order to attain this object, an incircuit emulator concerning this invention is constituted so that it may have memory in which read/write is possible by control of a function to stop a original oscillation in a mode of operation of a microcomputer, control software of an incircuit emulator for debugging, and control software.
[0013] According to said configuration, after termination of interrupt processing generated in the state of a stop mode of a user program According to an input of an rerun instruction, a return address which shunted to a stack area is rewritten to a dummy return address. An instruction put into a field of a dummy return address at a stop mode and an instruction which carries out an unconditional branch to a field of degree address to a return address of a main routine are written in. An instruction put into a stop mode can be executed and it can return to a condition of a stop mode before a user's not having any constraint and stopping a user program.
[0014]
[Embodiment of the Invention] Hereafter, the gestalt of the operation in this invention is explained to details with reference to a drawing. Drawing 1 is drawing showing going into the original stop mode, when ICE interrupt processing is performed into the stop mode of the user program in the gestalt of this

operation and a user program is rerun after that. Moreover, drawing 2 is a flow chart which shows processing actuation of the control software of ICE in the gestalt of operation of this invention. They are drawing showing how to use the stack for putting drawing 3 into a stop mode by rerun of a user program after ICE interrupt processing, and drawing showing the instruction of each address which drawing 4 shifted to the stop mode in the user program, and changed into the standby condition.
[0015] In drawing 3 the shunting field of the flag status of 12 and the various statuses of 14 They are the field which shunts the flag status automatically the same with having been shown in drawing 8 at the time of interruption generating, and the field which shunts the various statuses. 15 The dummy return address which rewrote the return address 13 of the program of drawing 8 with the control software of ICE, and changed it, Moreover, the instruction to which 16 is put into the stop mode rewritten by the control software of ICE, and 17 are instructions which carry out an unconditional branch to the return address of a main routine. Moreover, in drawing 4 , the 2nd instruction with which the 1st instruction immediately after the instruction put into the stop mode which 19 has in a user program, and the instruction 19 to which 20 is put into a stop mode, and 21 are most performed first when a stop mode is canceled by a user's interruption, and 22 are the waiting for an interruption return.
[0016] First, explanation of operation in the condition of having carried out an activation halt of the user program is performed by ICE interrupt processing generated in the stop mode of a user program using drawing 3 and drawing 4 . If the instruction 19 put into the Ath stop mode in the main routine of the user program shown in drawing 4 is executed, since the next instruction is also prefetched (prefetch), the Bth instruction [1st] 20 will be executed. Till this event, it is normal mode, and an oscillation stops immediately after executing the Bth instruction [1st] 20, and it becomes a stop mode. In this stop mode, the input of waiting for interruption return 22 is stood by, and the return to the main routine by the interruption return from a user is performed from the instruction 21 of the 2nd of an address C address, and serves as normal mode.
[0017] Here, in the standby condition of waiting for interruption return 22 in a stop mode, if ICE interrupt processing of the control software of ICE occurs in order to debug by performing halt actuation of a user program, a stop mode will be canceled like interrupt processing of a user program, and will turn into normal mode. It shunts, as the condition of the stack area at this time is shown in drawing 8 , and to the return address 13 of drawing 8 , the address C address of the 2nd instruction 21 shown in drawing 4 shunts.
[0018] Furthermore, actuation of the control software of ICE is explained using the flow chart of drawing 2 . It checks whether the user program has stopped during activation of a user program according to generating of activation halt processings (processing of interruption by compulsive breaking etc.) (S1). In processing S1, a check of that the user program stopped according to generating of activation halt processing reads the modes of operation (normal mode, stop mode, etc.) of the microcomputer before going into the activation halt processing of a user program (S2). Furthermore, it is read from a general-purpose register according to generating of interruption by ICE etc., and reading processing of status information in which it shunted to the stack area is performed (S3), and processing of the generated interruption is performed. Then, it is checked whether the return instruction of processing of said interruption and the instruction of rerun of the user program by actuation of a user have been issued (S4).
[0019] In the aforementioned processing S4, when an instruction of rerun is checked, it checks whether the condition before a halt of a user program is a stop mode from the information read by said processing S2 (S5). When it is not a stop mode in this processing S5, processing of rerun of a user program is started by executing the return instruction which performs writing for the status information of the address and the status which shunted to the stack area to a general-purpose register (S6).
[0020] Moreover, processing which writes in the instruction which performs processing which changes into a dummy return address the return address which has shunted to the stack area at the time of a stop mode (S7), writes in the instruction put into the field specified as a dummy return address at a stop mode (S8), and carries out an unconditional branch to the return address of the normal of a user program further in processing S5 is performed (S9). After the above processings, a return instruction is executed

and processing of rerun of a user program is started (S6).
[0021] When making a user program specifically rerun by the actuation after ICE interrupt processing generated in the condition of a stop mode and considering as the same stop mode as interruption before, the following change is made with the control software of ICE according to the check of the rerun instruction of processing S4 shown in drawing 2 . First, it checks whether the mode of operation before a halt of a user program is a stop mode by processing S5. Although C address which is a return address to the main routine shown in drawing 4 is stored in the return address 13 (refer to drawing 8) which has shunted to the Yth street of a stack area in processing S7 if it is a stop mode, it changes into the Vth street as a dummy return address 15. Next, the instruction 16 put into a stop mode is written in the Vth street of a stack area by processing S8, and the instruction which branches to C address shown in drawing 4 as instruction 17 which carries out an unconditional branch to the following street [Wth] to the return address of a main routine by processing S9 is written in.
[0022] Where such pretreatment is performed, if the ICE return instruction of ICE interrupt processing is executed, a microcomputer will return automatically X address of the stack area in drawing 3 , the Yth flag status [Zth] 12, the dummy return address 15, and the various statuses 14 to a general-purpose register, and rerun of a user program will be performed. And a user program will be performed from the Vth street which is the dummy return address 15, the instruction 16 put into a stop mode is executed, and it goes into a stop mode. Since an oscillation stops after the Wth activation which is the next instruction, a user can operate ICE [in / in a user program / a stop mode], without paying any attentions.
[0023] In addition, the Vth instruction [Wth] of a stack area must use the instruction from which the flag status 12 and the various statuses 14 do not change. Moreover, in the above explanation, although explained using a stack area, even if it uses the memory in which the read/write of a free space is possible, it can carry out similarly.
[0024]
[Effect of the Invention] As explained above, according to this invention, activation of ICE in the stop mode of a user program and actuation of a halt are attained without applying the burden on actuation to a user, and the effect that it can debug efficiently is done so.

[Translation done.]

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-030478
(43)Date of publication of application : 02.02.1996

(51)Int.CI. G06F 11/22
G06F 11/28

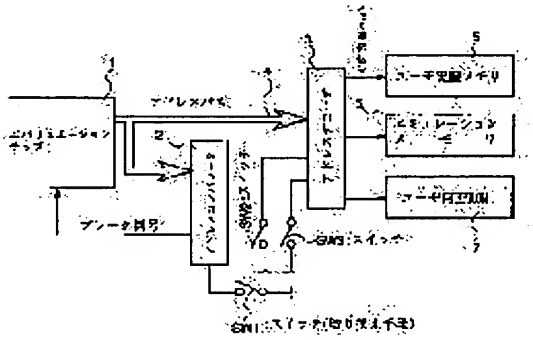
(21)Application number : 06-167634
(22)Date of filing : 20.07.1994

(71)Applicant : HITACHI LTD
(72)Inventor : SANO RYOICHI
YOSHIOKA KEIKO

(54) EMULATOR

(57)Abstract:
PURPOSE: To improve the real-time performance to an application system at the start of a user program right after the power-ON resetting or forcible resetting of an emulator.

CONSTITUTION: A vector fetch address, a reset vector address, and data of a switch SW1 are inputted to an address comparator 2, the end of the resetting of an evaluation chip 1 is confirmed with the vector fetch address and reset vector address, and whether or not a break signal is outputted is optionally set according to the setting of the switch SW1. When the switch SW is OFF, no signal is outputted to the address comparator 2 and the break signal is outputted to start firmware. When the switch SW1 is ON, on the other hand, a signal is outputted to the address comparator 2 and the break signal is not inputted to the evaluation chip 1; and the firmware is not started and the user program is started.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平8－30478

(43)公開日　平成８年(1996)２月２日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ	11/22	３４０　Ａ			
	11/28		Ｌ　7313－5Ｂ		

審査請求　未請求　請求項の数４　ＯＬ　（全９頁）

(21)出願番号　特願平6－167634

(22)出願日　平成６年(1994)７月20日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　佐野　亮一
東京都小平市上水本町５丁目20番１号　株式会社日立製作所半導体事業部内

(72)発明者　吉岡　桂子
東京都小平市上水本町５丁目20番１号　株式会社日立製作所半導体事業部内

(74)代理人　弁理士　筒井　大和

(54)【発明の名称】　エミュレータ

(57)【要約】

【目的】　エミュレータのパワーオンリセットまたは強制リセット直後のユーザプログラム起動における応用システムに対するリアルタイム性を向上させる。

【構成】　アドレスコンパレータ２にベクタフェッチアドレス、リセットベクタアドレスおよびスイッチＳＷ１のデータを入力し、ベクタフェッチアドレス、リセットベクタアドレスによりエバリュエーションチップ１のリセット終了を確認し、スイッチＳＷ１の設定によってブレーク信号を出力するか否かを任意に設定する。スイッチＳＷ１が非導通の場合、アドレスコンパレータ２に信号は出力されず、ブレーク信号が出力されファームウェアを起動させる。スイッチＳＷ１が導通の場合、アドレスコンパレータ２に信号が出力され、エバリュエーションチップ１はブレーク信号が入力されずファームウェアは起動させずにユーザプログラムを起動させる。

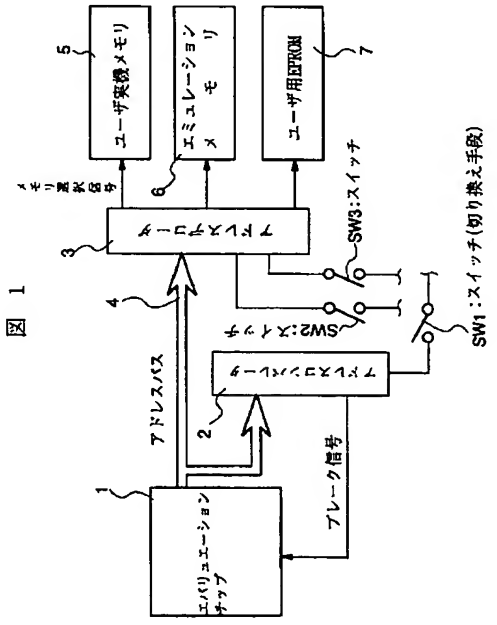


図１

1

【特許請求の範囲】

【請求項１】　マイクロコンピュータを用いた応用システムのソフトウェアおよびハードウェアの評価を行うエミュレータであって、ターゲットマイクロコンピュータにおける機能の代行およびデバッグ機能の制御を行うエバリュエーションチップのリセット直後において、前記エバリュエーションチップにブレーク信号を出力するブレーク信号発生手段を設け、前記ブレーク信号発生手段から出力されたブレーク信号に基づいてファームウェアまたはユーザプログラムを選択し、起動させることを特徴とするエミュレータ。

【請求項２】　前記ブレーク信号発生手段が、ファームウェアまたはユーザプログラムのどちらか一方を選択する所定の信号を出力する切り換え手段と、前記エバリュエーションチップのリセット直後において、プログラムの開始アドレスを指定するベクタフェッチアドレスと前記エバリュエーションチップがアクセスするリセットベクタアドレスと前記切り換え手段から入力される所定の信号との比較結果に基づいてブレーク信号を発生するアドレスコンパレータとよりなることを特徴とする請求項１記載のエミュレータ。

【請求項３】　前記切り換え手段が、エバリュエーションチップから出力される制御信号を一時的に記憶し、前記アドレスコンパレータに出力するレジスタよりなることを特徴とする請求項２記載のエミュレータ。

【請求項４】　前記ブレーク発生手段が、前記切り換え手段からの信号入力と前記エバリュエーションチップのリセット信号との論理積を出力する論理積回路と、前記論理積回路より出力された所定の信号をラッチした後に出力するラッチ回路とよりなり、前記切り換え手段からの入力信号および前記エバリュエーションチップのリセット信号を前記論理積回路に入力し、前記論理積回路から出力された信号を前記ラッチ回路によりラッチし、前記エバリュエーションチップのリセットが終了するとブレーク信号として前記エバリュエーションチップに入力することを特徴とする請求項１または３記載のエミュレータ。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、マイクロプロセッサを用いた応用システムの開発支援装置であるエミュレータに関し、特に、ターゲットマイコンにおける機能の代行およびエミュレータの制御を１種類のマイコンにより行うエミュレータにおけるプログラムの起動に適用して有効な技術に関するものである。

【０００２】

【従来の技術】本発明者が検討したところによれば、ターゲットマイコンにおける機能の代行およびエミュレータの制御を１種類のマイコンにより行う、いわゆる１ＣＰＵ方式のエミュレータでは、電源投入時のリセット

2

（以下、パワーオンリセットという）またはリセットスイッチによるリセット（以下、強制リセットという）後におけるユーザプログラムの起動は、ファームウェアを起動させた後に行われいた。

【０００３】また、本発明者の検討によれば、この動作の流れは、図５に示すように、パワーオンリセットまたは強制リセット時（ステップ３０１）において、エミュレータを動作させるソフトウェアである、いわゆるファームウェアを起動（ステップ３０２）させ、予め設定されているスイッチによってファームウェアの動作続行またはユーザプログラムの起動を選択（ステップ３０３）している。

【０００４】なお、エミュレータについて記述されている例としては、日立マイクロコンピュータエンジニアリング株式会社発行「日立マイコン技法」１９８８年Ｖｏｌ．２Ｎｏ．２、Ｐ２１～Ｐ２７がある。

【０００５】

【発明が解決しようとする課題】ところが、上記のようなパワーオンリセットまたは強制リセット時のエミュレータにおける初期化では、次のような問題点があることが本発明者により見い出された。

【０００６】すなわち、評価が行われるユーザの応用システムが複数のＣＰＵにより構成されたマルチＣＰＵ方式であると、エミュレーションの初期化が行われている間に、応用システムに搭載されているターゲットＣＰＵ以外の他のＣＰＵが立ち上がってしまい、プログラムが動作してしまうことになる。

【０００７】それによって、エミュレータと応用システムに搭載されているＣＰＵとの同期性がとれなくなってしまい、応用システムの実動作に対するリアルタイム性が損なわれてしまう。

【０００８】本発明の目的は、エミュレータのパワーオンリセットまたは強制リセット直後のユーザプログラム起動における応用システムの実動作に対するリアルタイム性を向上させるエミュレータを提供することにある。

【０００９】本発明の前記ならびにその他の目的と新規な特徴は、本明細書の記述および添付図面から明らかになるであろう。

【００１０】

【課題を解決するための手段】本願において開示される発明のうち、代表的なものの概要を簡単に説明すれば、以下のとおりである。

【００１１】本発明のエミュレータは、ターゲットマイクロコンピュータにおける機能の代行およびデバッグ機能の制御を行うエバリュエーションチップのリセット直後において、エバリュエーションチップに任意のブレーク信号を出力するブレーク信号発生手段を設け、ブレーク信号発生手段から出力されたブレーク信号に基づいてファームウェアまたはユーザプログラムを選択し、起動させるものである。

【００１２】また、本発明のエミュレータは、前記ブレーク信号発生手段が、ファームウェアまたはユーザプログラムのどちらか一方を選択する所定の信号を出力する切り換え手段と、エバリュエーションチップのリセット直後において、プログラムの開始アドレスを指定するベクタフェッチアドレスとエバリュエーションチップがアクセスするリセットベクタアドレスと切り換え手段から入力される所定の信号との比較結果に基づいてブレーク信号を発生するアドレスコンパレータとよりなるものである。

【００１３】さらに、本発明のエミュレータは、前記切り換え手段が、エバリュエーションチップから出力される制御信号を一時的に記憶し、アドレスコンパレータに出力するレジスタよりなるものである。

【００１４】また、本発明のエミュレータは、前記ブレーク発生手段が、切り換え手段からの信号入力とエバリュエーションチップのリセット信号との論理積を出力する論理積回路と、論理積回路より出力された所定の信号をラッチした後に出力するラッチ回路とよりなり、切り換え手段からの入力信号およびエバリュエーションチップのリセット信号を論理積回路に入力し、論理積回路から出力された信号をラッチ回路によりラッチし、エバリュエーションチップのリセットが終了するとブレーク信号としてエバリュエーションチップに入力するものである。

【００１５】

【作用】上記した本発明のエミュレータによれば、エバリュエーションチップのリセット直後において、ブレーク信号発生手段によってファームウェアまたはユーザプログラムのどちらを起動するかを任意に選択できるブレーク信号を、エバリュエーションチップに入力することができる。

【００１６】また、上記した本発明のエミュレータによれば、切り換え手段によりファームウェアまたはユーザプログラムのどちらか一方を選択する所定の信号を出力させ、切り換え手段により出力された所定の信号とエバリュエーションチップのリセット後のプログラムの開始アドレスを指定するベクタフェッチアドレスとエバリュエーションチップがアクセスするリセットベクタアドレスとをアドレスコンパレータに入力させることにより比較させ、その比較結果に基づいてブレーク信号を発生させることによって、リセット直後に確実に任意のブレーク信号を発生させることができる。

【００１７】さらに、上記した本発明のエミュレータによれば、切り換え手段をレジスタとすることによってエバリュエーションチップから出力される制御信号に基づいてユーザがソフトウエアにより、ファームウェアまたはユーザプログラムの起動を選択することができる。

【００１８】また、上記した本発明のエミュレータによれば、論理積回路に、切り換え手段からの信号入力とエバリュエーションチップのリセット信号とを入力し、その論理積をラッチ回路によりラッチし、エバリュエーションチップのリセットが終了するとブレーク信号としてエバリュエーションチップに入力することにより、より簡単な回路構成によってファームウェアまたはユーザプログラムのどちらを起動するかを選択することができる。

【００１９】それにより、エミュレータのパワーオンリセットまたは強制リセット後において、ユーザプログラムを起動させる場合にファームウェアを起動させることなくユーザプログラムを起動させることができ、エミュレータと応用システムに搭載されているＣＰＵとのリアルタイム性を損なわずにデバッグを行うことができる。

【００２０】

【実施例】以下、本発明の実施例を図面に基づいて詳細に説明する。

【００２１】（実施例１）図１は、本発明の実施例１によるエミュレータのブレーク信号発生回路周辺における要部機能ブロック図、図２は、本発明の実施例１によるエミュレータにおけるブレーク信号発生の説明図である。

【００２２】本実施例１において、ソフトウェアおよびハードウェアのデバッグ、評価を行うエミュレータは、ターゲットマイクロコンピュータの機能を代行およびデバッグ機能の制御を行うマイクロコンピュータであるエバリュエーションチップ１が設けられている。

【００２３】また、エバリュエーションチップ１は、所定のアドレスデータを比較し、その比較結果が所定の結果となると信号を発生するアドレスコンパレータ２および入力信号の組合せから所定のメモリを選択するアドレスデコーダ３がアドレスバス４を介して接続されている。

【００２４】さらに、アドレスコンパレータ２から出力される信号は、エバリュエーションチップ１の所定の入力ピンにブレーク信号として入力されるように接続されている。

【００２５】また、アドレスデコーダ３によって選択されるメモリには、ユーザが開発中のマイクロコンピュータを用いた応用システム（図示せず）に搭載されているユーザ実機メモリ５、エミュレータに設けられ、ユーザ実機メモリ５の機能を代行するエミュレーションメモリ６およびユーザ用ＥＰＲＯＭ７がある。

【００２６】さらに、アドレスコンパレータ２には、エバリュエーションチップ１のリセット後にファームウェアを起動させるかユーザプログラムを起動させるかの選択を行うスイッチ（切り換え手段）ＳＷ１が接続されている。

【００２７】また、アドレスデコーダ３には、ユーザ実機メモリ５、エミュレーションメモリ６またはユーザ用ＥＰＲＯＭ７をアドレスデコーダ３に選択させるスイッ

チＳＷ２，ＳＷ３が接続され、それぞれのスイッチＳＷ１～ＳＷ３が導通状態となると信号が入力される。

【００２８】さらに、これらスイッチＳＷ１～ＳＷ３は、たとえばショートバースイッチなどの機械的なスイッチに構成されている。

【００２９】次に、本実施例の作用について説明する。

【００３０】まず、エミュレータに電源が投入され、パワーオンリセットが終了すると、プログラム開始のアドレスを指定するためのＨｉ信号またはＬｏ信号の一方における固定入力値であるベクタフェッチアドレスがアドレスコンパレータ２に入力される。

【００３１】また、その時、エバリュエーションチップ１がデータバス（図示せず）を介してアクセスするリセットベクタアドレスおよびスイッチＳＷ１の信号が同時にアドレスコンパレータ２に入力される。

【００３２】ここで、本実施例において、たとえば、スイッチＳＷ１が導通状態となり所定の信号出力されている、すなわちＨｉ信号がアドレスコンパレータ２に入力されている場合にユーザプログラムを起動させ、スイッチＳＷ１が非導通状態となり所定の信号が出力されない、すなわちＬｏ信号がアドレスコンパレータ２に入力されている場合にはファームウェアを起動させるものとする。

【００３３】まず、ユーザプログラムを起動させる場合について説明する。

【００３４】ユーザがエミュレータの電源を投入する以前にスイッチＳＷ１を導通状態となるように設定する。アドレスコンパレータ２は、ベクタフェッチアドレス、リセットベクタアドレスおよびスイッチＳＷ１の入力信号を比較することによってエミュレータの立ち上がり時であることならびにユーザプログラムを起動させることを判断する。

【００３５】よって、ユーザプログラムを起動させるので、アドレスコンパレータ２は、ブレーク信号をエバリュエーションチップ１に出力しない。この時、同時に、ユーザによってエミュレータに電源投入を行う以前に設定されたスイッチＳＷ２，ＳＷ３の導通状態を示す信号がアドレスデコーダにも入力される。

【００３６】そして、スイッチＳＷ２，ＳＷ３の信号入力の状態によって、アドレスデコーダ３がユーザ実機メモリ５、エミュレーションメモリ６およびユーザ用ＥＰＲＯＭ７のそれぞれを選択する。

【００３７】また、本実施例では、たとえば、スイッチＳＷ２が非導通状態、すなわちアドレスデコーダ３にスイッチＳＷ２からの信号入力がない場合にユーザ実機メモリ５を選択し、スイッチＳＷ２が導通状態となり、スイッチＳＷ３が非導通状態、すなわちアドレスデコーダ３にスイッチＳＷ３からの信号入力がない場合にエミュレーションメモリ６を選択し、スイッチＳＷ３が導通状態となるとユーザ用ＥＰＲＯＭ７を選択するものとする。

【００３８】たとえば、エミュレーションメモリ６を選択すると、スイッチＳＷ２は導通状態、スイッチＳＷ３は非導通状態となり、アドレスデコーダ３によってエミュレーションメモリ６が選択され、エミュレーションメモリ６に保持又は格納されているユーザプログラムが起動することになる。

【００３９】よって、エミュレータの電源立ち上げによるリセット後に、エバリュエーションチップ１にはブレーク信号が入力されないことになり、ファームウェアの立ち上げが行われず、ユーザが設定したスイッチＳＷ２，ＳＷ３の導通状態によりアドレスデコーダ３がユーザ実機メモリ５，エミュレーションメモリ６，ユーザ用ＥＰＲＯＭ７を選択し、ユーザプログラムが起動される。

【００４０】次に、ファームウェアを起動させる場合について説明する。

【００４１】ユーザがエミュレータ２の電源を投入する以前にスイッチＳＷ１を非導通状態となるように設定する。アドレスコンパレータ２は、ベクタフェッチアドレス、リセットベクタアドレスおよびスイッチＳＷ１の入力信号を比較することによってエミュレータの立ち上がり時であることならびにファームウェアを起動させることを判断する。

【００４２】このファームウェアを起動させる場合、アドレスコンパレータ２はブレーク信号を発生させ、エバリュエーションチップ１に出力する。また、ファームウェア起動時には、ユーザによってエミュレータに電源投入を行う以前に設定されたスイッチＳＷ２，ＳＷ３のどのような状態であっても無効となる。

【００４３】そして、エミュレータの電源立ち上げによるリセット時に、アドレスコンパレータ２から出力されたブレーク信号がエバリュエーションチップ１に入力されるので、ファームウェアの立ち上げが行われ、初期化される。

【００４４】よって、これらの信号の流れは、図２に示すように、電源投入によりパワーオンリセットが開始（ステップ１０１）されると、スイッチＳＷ１が導通状態（ステップ１０２）では、ブレーク信号は出力されない。

【００４５】そして、スイッチＳＷ２が非導通状態であると、ユーザ実機メモリが選択（ステップ１０３）され、スイッチＳＷ２が導通状態であり、スイッチＳＷ３が非導通状態であるとエミュレーションメモリが選択（ステップ１０４）され、スイッチＳＷ３が導通状態であるとユーザ用ＥＰＲＯＭが選択（ステップ１０５）される。

【００４６】次に、スイッチＳＷ１が非導通状態であると、ブレーク信号が出力（ステップ１０５）され、ファームウェアが起動（ステップ１０６）となる。そして、

初期化が終了すると、ファームウェアの連続続行状態（ステップ１０７）となる。

【００４７】また、ユーザプログラムからファームウェアへの切り換えは、エミュレータ外部に設けられたデータ入出力手段（図示せず）により、ブレーク命令を入力することによって切り換え、ファームウェアからユーザプログラムへの切り換えは当該データ入出力手段によりＲＴＢ（Ｒｅｔｕｒｎ　Ｔｏ　Ｂｒｅａｋ）命令を入力することによって行う。

【００４８】さらに、本実施例では、電源投入時のエミュレータ立ち上げにおけるパワーオンリセット時について記載したが、たとえば、ユーザがリセットスイッチにより強制的にリセットする強制リセット時についても動作は同じである。

【００４９】それにより、本実施例１では、ユーザがスイッチ１～３を予め任意に設定しておくことによって、エミュレータの立ち上げ時にファームウェアを起動させずにユーザプログラムを起動させることができ、応用システムの実動作に対するリアルタイム性を損なわずにデバッグを行うことができる。

【００５０】（実施例２）図３は、本発明の実施例２によるエミュレータのブレーク信号発生回路周辺における要部機能ブロック図である。

【００５１】本実施例２においては、エミュレータの立ち上がり時にファームウェアを起動させるかユーザプログラムを起動させるかの選択を行う信号をレジスタ８によってアドレスコンパレータ２に入力する。

【００５２】また、このレジスタ８には、電源が遮断されても情報を保持するバックアップ８ａが設けられ、レジスタ８に入力される制御信号はエバリュエーションチップ１から出力される。

【００５３】さらに、この場合においても、レジスタ８からアドレスコンパレータ２に出力される信号がＬｏ信号であればユーザプログラムを起動させ、Ｈｉ信号であればファームウェアを起動させるとする。

【００５４】ここで、ユーザプログラムを起動させる場合について説明する。

【００５５】まず、リセットスイッチによる強制リセットによりユーザプログラムを起動させる場合であると、強制リセットを行う前に、ユーザはデータ入出力手段（図示せず）のキーボードから強制リセット時にユーザプログラムを起動させるためのデータを入力する。そして、そのデータがエバリュエーションチップ１に入力されると、エバリュエーションチップ１はレジスタ８にＬｏ信号を入力し、レジスタ８にデータを記憶させる。

【００５６】次に、ユーザはリセットボタンを押し、強制リセットを行う。この時、アドレスコンパレータ２には、ベクタフェッチアドレス、リセットベクタアドレスおよびレジスタ８に記憶されているＬｏ信号が入力される。

【００５７】よって、前記実施例１と同様に、レジスタ８からの信号はＬｏであるので、ファームウェアの立ち上げは行われず、ユーザが設定したスイッチＳＷ２，ＳＷ３の導通状態により、アドレスデコーダ３がユーザ実機メモリ５，エミュレーションメモリ６，ユーザ用ＥＰＲＯＭ７を選択し、ユーザプログラムが起動される。

【００５８】次に、リセットスイッチによる強制リセットによりファームウェアを起動させる場合であると、強制リセットを行う前に、ユーザはデータ入出力手段のキーボードから強制リセット時にファームウェアを起動させるためのデータを入力する。　そして、そのデータがエバリュエーションチップ１に入力されると、エバリュエーションチップ１はレジスタ８にＨｉ信号を入力し、記憶させる。

【００５９】ユーザによりリセットボタンが押され、強制リセットとなると、アドレスコンパレータ２には、ベクタフェッチアドレス、リセットベクタアドレスおよびレジスタ８に記憶されているＨｉ信号が入力される。

【００６０】よって、レジスタ８から出力された信号がＨｉ信号であるので、アドレスコンパレータ２はブレーク信号をエバリュエーションチップ１に出力し、ファームウェアが起動される。

【００６１】また、この場合も、ファームウェア起動時には、ユーザによってエミュレータ２に電源投入を行う以前に設定されたスイッチＳＷ２，ＳＷ３のどのような状態であっても無効となる。

【００６２】さらに、電源投入時のパワーオンリセットを行うときも、予めキーボードから所定のデータを入力し、レジスタ８に記憶させる。

【００６３】また、この場合、電源が遮断されてもバックアップ８ａによりデータは保持されているので、本実施例に記載した強制リセットと同じ動作により、ユーザプログラムの起動またはファームウェアの起動を選択することができる。

【００６４】それにより、本実施例２によれば、ユーザプログラムまたはファームウェアの起動による設定をレジスタ８にデータ保持できるので、キーボードによりデータを入力することができ、遠隔操作による設定も可能となる。

【００６５】（実施例３）図４は、本発明の実施例３によるエミュレータのブレーク信号発生回路周辺における要部機能ブロック図である。

【００６６】本実施例３においては、パワーオンリセットまたは強制リセット時に出力されるリセット信号の入力とスイッチＳＷ４（切り換え手段）の入力信号との論理積を出力するＡＮＤ回路（論理積回路）９および当該ＡＮＤ回路９から出力された信号をラッチした後にエバリュエーションチップ１にブレーク信号として出力するラッチ回路１０が設けられてる。

【００６７】また、このラッチ回路１０の出力信号は、

エバリュエーションチップ１のブレーク信号が入力される所定の入力ピンに入力されるように接続される。さらに、本実施例では、リセット時のリセット信号は、たとえばアクティブ・ハイとする。

【００６８】まず、ここで、ファームウェアを起動させる場合であると、ユーザはエミュレータに電源を投入する以前にスイッチＳＷ４を導通状態としておく。

【００６９】そして、エミュレータに電源が投入され、パワーオンリセットによるリセット信号が出力されると、ＡＮＤ回路９の一方の入力には、リセット信号のＨｉ信号が入力される。また、スイッチＳＷ４が接続されているＡＮＤ回路９の他方の入力にも、スイッチＳＷ４が導通状態となっているのでＨｉ信号が入力される。

【００７０】よって、ＡＮＤ回路９の入力はどちらも、Ｈｉ信号となるので、その論理積出力はＨｉ信号となる。

【００７１】次に、ＡＮＤ回路９から出力されたＨｉ信号は、その後段に接続されているラッチ回路１０に入力される。そして、このラッチ回路１０に入力されたＨｉ信号は、エバリュエーションチップ１のリセットが終了するまでラッチされ、エバリュエーションチップ１のリセットが終了するとラッチ回路１０からエバリュエーションチップ１の所定の入力ピンにブレーク信号として出力される。

【００７２】よって、前記実施例１と同様に、エミュレータの電源立ち上げによるリセット時に、アドレスコンパレータ２から出力されたブレーク信号がエバリュエーションチップ１に入力されるので、ファームウェアの立ち上げが行われ、初期化される。

【００７３】また、この場合も、ファームウェア起動時には、ユーザによってエミュレータに電源投入を行う以前に設定されたスイッチＳＷ２，ＳＷ３のどのような状態であっても無効となる。

【００７４】次に、ユーザプログラムを起動させる場合であると、ユーザはエミュレータに電源を投入する以前にスイッチＳＷ４を非導通状態としておく。

【００７５】エミュレータに電源が投入され、リセット信号が出力されると、ＡＮＤ回路９の一方の入力には、リセット信号のＨｉ信号が入力され、スイッチＳＷ４が接続されているＡＮＤ回路９の他方の入力には、スイッチＳＷ４が非導通状態となっているので、Ｌｏ信号が入力される。

【００７６】よって、ＡＮＤ回路９の一方の入力はＨｉ信号、他方の入力はＬｏ信号となり、出力はＬｏ信号となるので、ラッチ回路１０にもＬｏ信号がラッチされ、エバリュエーションチップ１のリセットが終了するとラッチ回路１０からの出力はＬｏ信号となりブレーク信号もＬｏ信号出力、すなわちブレーク信号が入力されないことになる。

【００７７】そして、パワーオンリセット時に、エバリュエーションチップ１にはブレーク信号が入力されないことになるのでファームウェアの立ち上げが行われない。

【００７８】また、ユーザが設定したスイッチＳＷ２，ＳＷ３の導通状態により、前記実施例１と同様に、アドレスデコーダ３がユーザ実機メモリ５，エミュレーションメモリ６，ユーザ用ＥＰＲＯＭ７を選択し、ユーザプログラムが起動される。

【００７９】さらに、本実施例でも、電源投入時のエミュレータ立ち上げにおけるパワーオンリセット時について記載したが、たとえば、ユーザがリセットスイッチにより強制的にリセットする強制リセット時についても動作は同じである。

【００８０】それにより、本実施例３では、より簡単な回路構成でエバリュエーションチップ１のリセット後にファームウェアを起動させずにユーザプログラムを起動させることができ、応用システムの実動作に対するリアルタイム性を損なわずにデバッグを行うことができる。

【００８１】以上、本発明者によってなされた発明を実施例に基づき具体的に説明したが、本発明は前記実施例に限定されるものではなく、その要旨を逸脱しない範囲で種々変更可能であることはいうまでもない。

【００８２】

【発明の効果】本願によって開示される発明のうち、代表的なものによって得られる効果を簡単に説明すれば、以下のとおりである。

【００８３】（１）本発明によれば、ブレーク信号発生手段から出力される任意のブレーク信号により、ファームウェアを起動させることなくユーザプログラムを起動させることができる。

【００８４】（２）また、本発明では、アドレスコンパレータにより、エバリュエーションチップのリセットが行われたか否かの確認を行うことによって、確実にリセット直後に任意のブレーク信号を生成できる。

【００８５】（３）さらに、本発明においては、レジスタを設けることによりユーザがソフトウエアによって任意のブレーク信号を設定できる。

【００８６】（４）また、本発明によれば、論理積回路およびラッチ回路を用いることにより、より簡単な回路構成において任意のブレーク信号を生成できる。

【００８７】（５）さらに、本発明では、上記（１）～（４）により、複数のＣＰＵにより構成されたマルチＣＰＵ方式の応用システムであっても、ＣＰＵと同じ時間でプログラムが起動するので、エバリュエーションチップのリセット直後におけるデバッグが行えるようになり、応用システムの実動作に対するリアルタイム性を損なわずにデバッグを行うことができる。

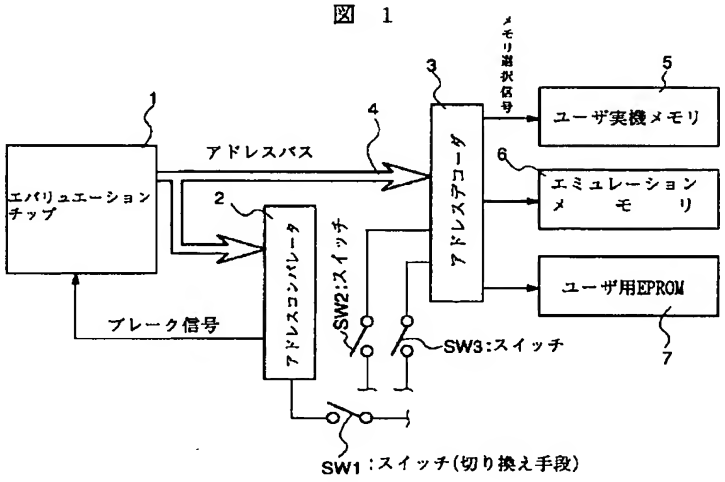
【図面の簡単な説明】

【図１】本発明の実施例１によるエミュレータのブレーク信号発生回路周辺における要部機能ブロック図であ

る。
【図２】本発明の実施例１によるエミュレータにおけるブレーク信号発生の説明図である。
【図３】本発明の実施例２によるエミュレータのブレーク信号発生回路周辺における要部機能ブロック図である。
【図４】本発明の実施例３によるエミュレータのブレーク信号発生回路周辺における要部機能ブロック図である。
【図５】本発明者により検討されたエミュレータのソフトウェア起動における説明図である。
【符号の説明】
１　エバリュエーションチップ
２　アドレスコンパレータ
＊３　アドレスデコーダ
４　アドレスバス
５　ユーザ実機メモリ
６　エミュレーションメモリ
７　ユーザ用ＥＰＲＯＭ
８　レジスタ
８ａ　バックアップ
９　ＡＮＤ回路（論理積回路）
１０　ラッチ回路
ＳＷ１　スイッチ（切り換え手段）
ＳＷ２　スイッチ
ＳＷ３　スイッチ
ＳＷ４　スイッチ（切り換え手段）

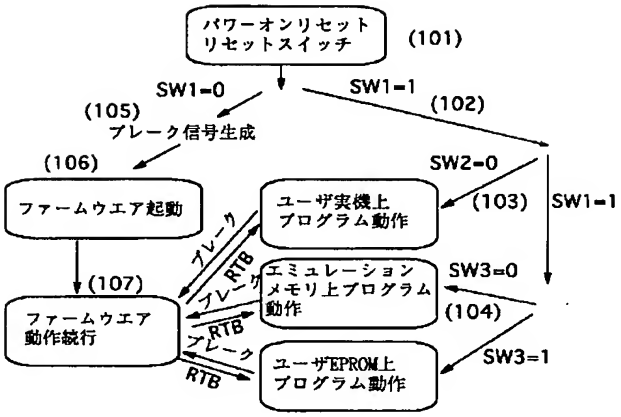
【図１】

図　１



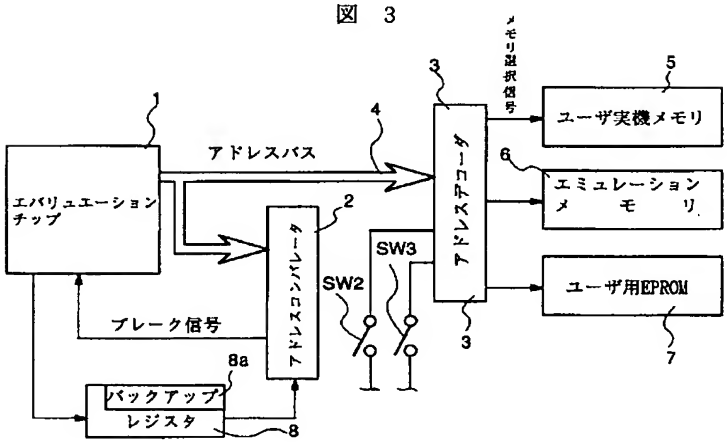
【図２】

図　2


パワーオンリセット
リセットスイッチ
(101)
SW1=0
SW1=1
(102)
(105)
ブレーク信号生成
(106)
SW2=0
ファームウエア起動
ユーザ実機上
プログラム動作
(103)
SW1=1
ブレーク
RTB
エミュレーション
メモリ上プログラム
動作
SW3=0
(104)
(107)
ファームウエア
動作続行
ブレーク
RTB
ユーザEPROM上
プログラム動作
SW3=1

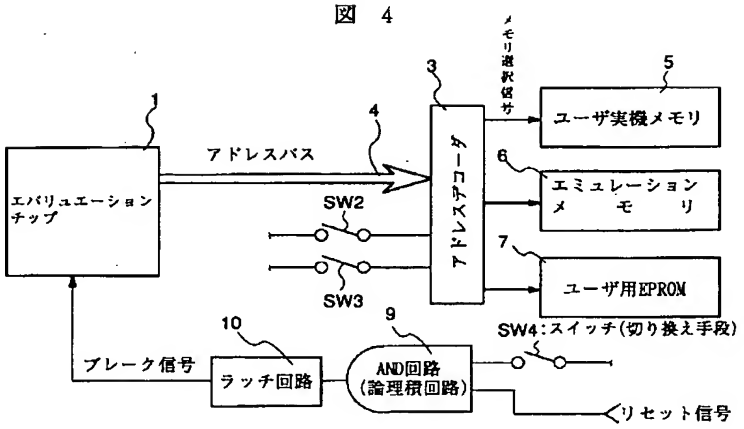
【図３】

図　3


エバリュエーション
チップ
1
アドレスバス
4
3
メモリ選択信号
5
ユーザ実機メモリ
6
エミュレーション
メ　モ　リ
2
アドレスコンパレータ
SW3
SW2
アドレスデコーダ
ユーザ用EPROM
3
7
ブレーク信号
8a
バックアップ
レジスタ
8

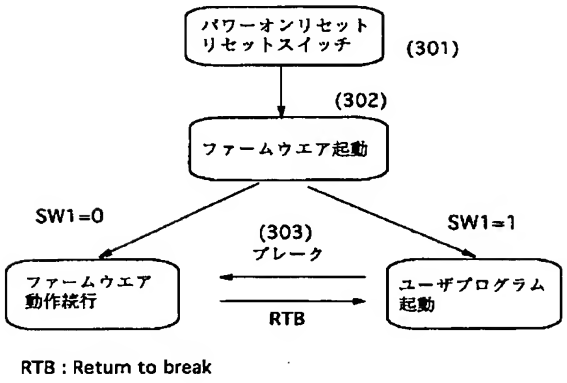
【図４】

図 4



【図５】

図 5



RTB : Return to break